RAPID UPDATE ON OPTICAL SIM + BEZIER TRACKS

Ben Jones

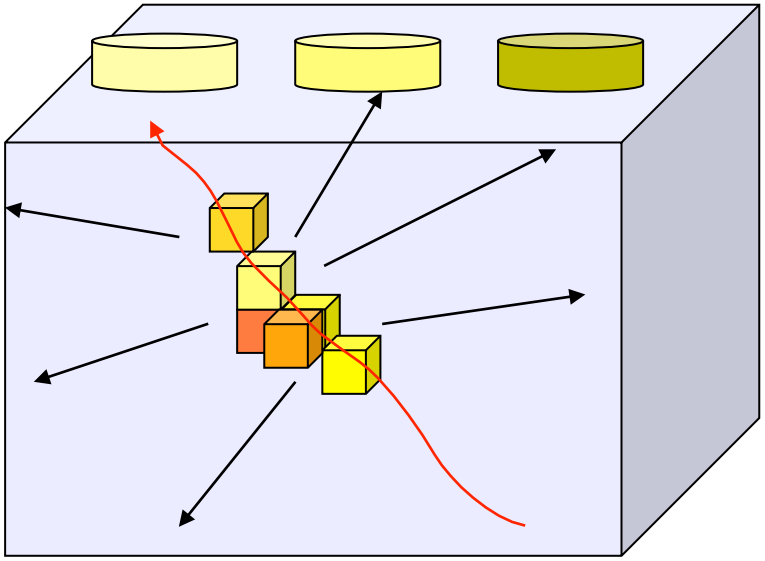
1. OPTICAL SIMULATIONS

PVS: Library Mode

- This service has now subsumed the old “PhotonLibraryService”, and handles all lookup based optical simulation methods
- As such these methods are now available to both simulation and reconstruction algorithms.
- This represents a MAJOR overhaul of the library building and sampling since the last talk

```
void SetVisibilityModel(int model) { fVisModel = model;  }
int GetVisibilityModel()           { return fVisModel;   }

double GetQuenchingFactor(double dQdx);

double DistanceToOpDet(                double* xyz, unsigned int OpChannel );
double SolidAngleFactor(               double* xyz, unsigned int OpChannel );
double GetVisibility(                  double* xyz, unsigned int OpChannel );

std::vector<double> GetAllVisibilities( double* xyz );

void StoreLibrary();


void StoreLightProd(    int  VoxID,  double  N );
void RetrieveLightProd( int& VoxID,  double& N );

void SetLibraryEntry(   int VoxID, int OpChannel, double N);
double GetLibraryEntry( int VoxID, int OpChannel);

bool IsBuildJob() { return fLibraryBuildJob; }

sim::PhotonVoxelDef GetVoxelDef() {return fVoxelDef; }
```

Library Building

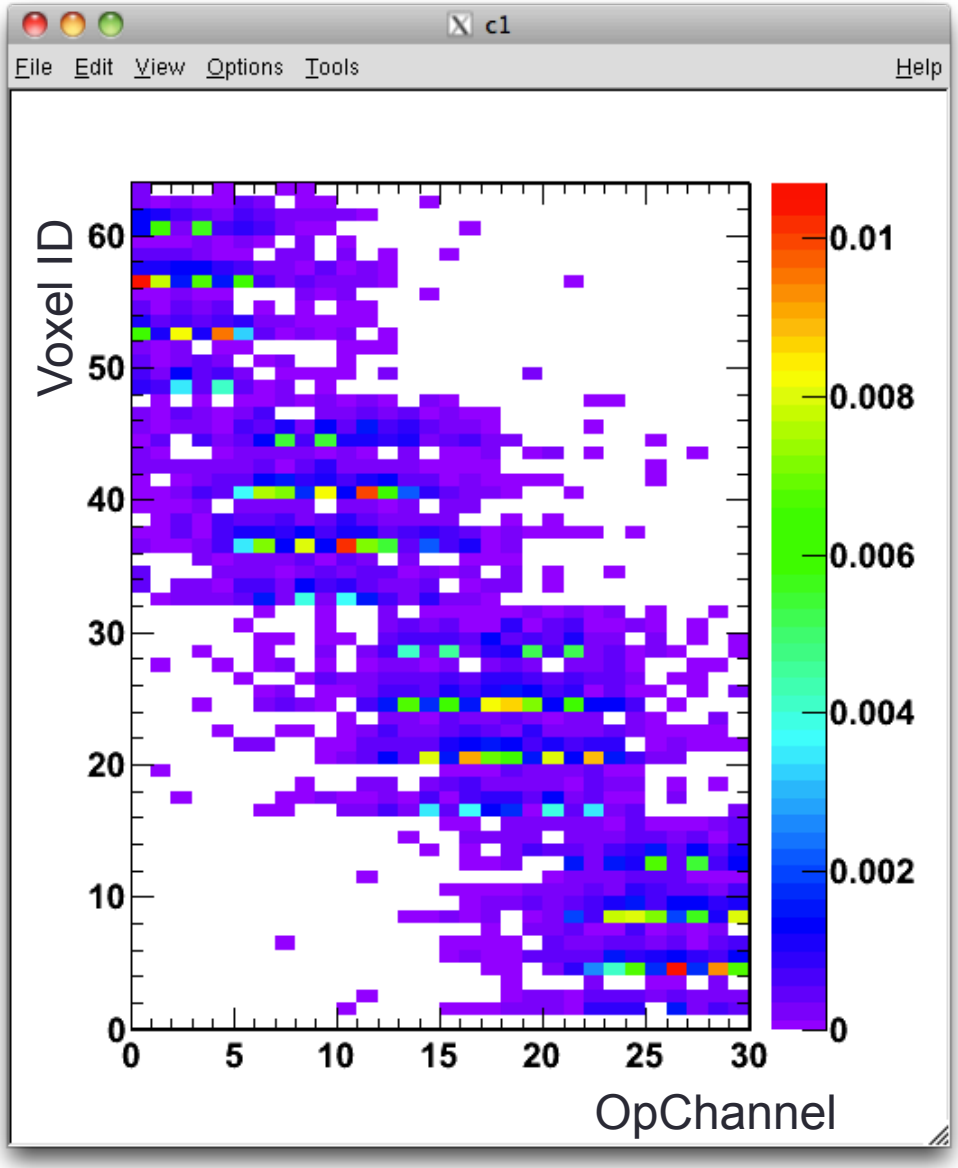


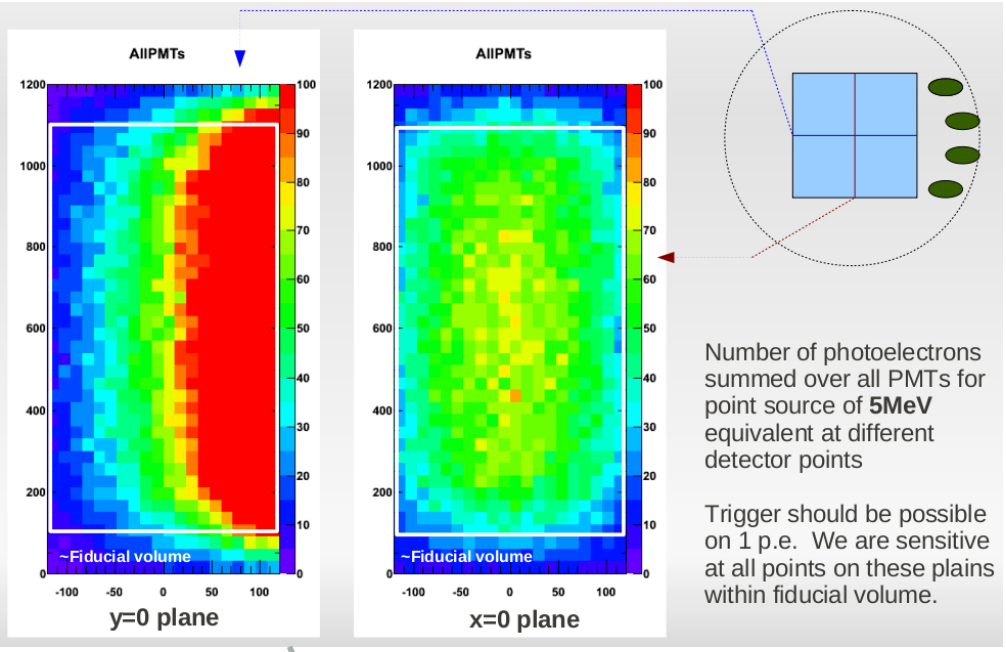
- Enable service PhotonVisibilityService
- Specify in its configuration:
  - *BuildLibrary : true*
  - *Number of voxels in x,y,z*
  - *Optional : sub-region of detector to simulate*
- Run modules LightSource, LArG4, SimPhotonCounter
- Brightness should be specified for LightSource. Other than this, PhotonVisibilityService will configure the modules appropriately.
- Library file is output through the TFileService

Library Build Test Run

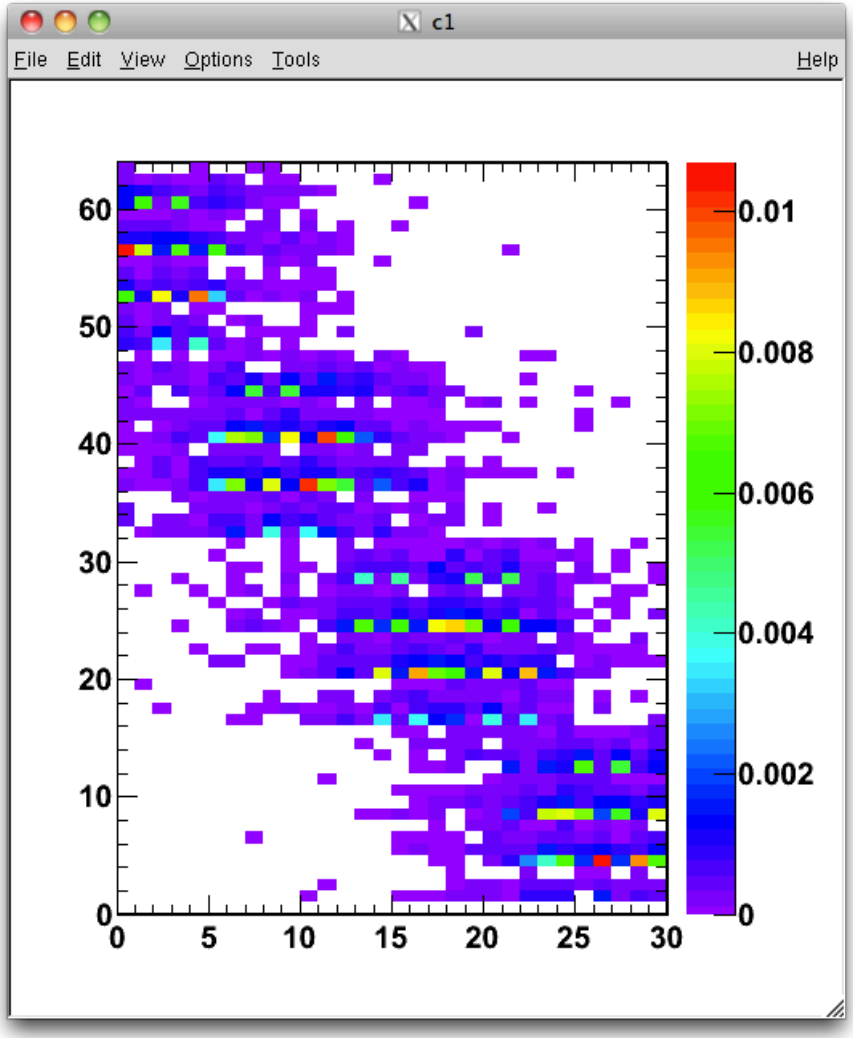
- Quick test run : 4x4x4 voxels, 10,000 photons in each
- Library build job takes about 10 mins

Colors show fraction of photons detected





Effectively the same information, but broken down per PMT and in a weird coordinate system.



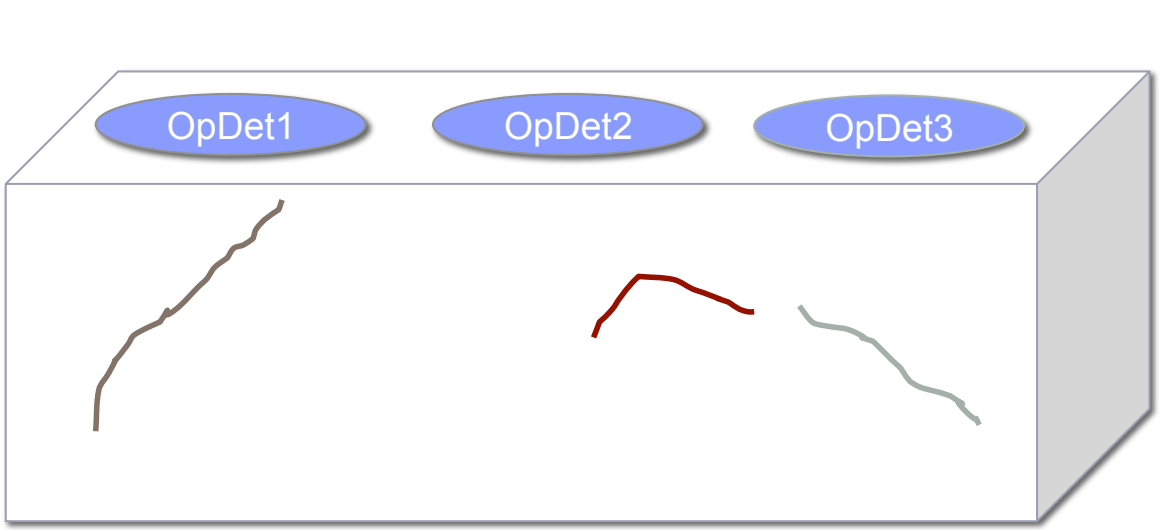
Library sampling simulation tool coming online soon (most code exist in uncommitted form).

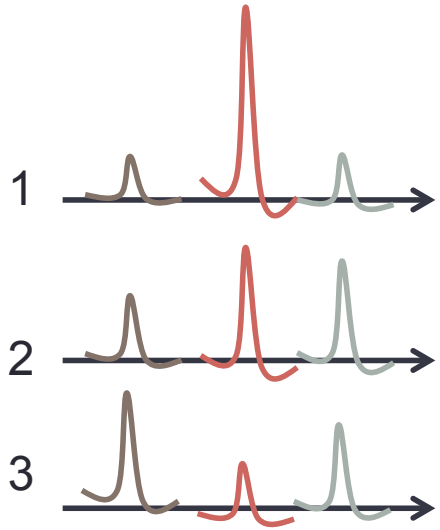
Scaling Up to Grid Job

- Need to get started building full MicroBooNE library using grid jobs.
- Do a few voxels in each and then combine library library files offline.
- Feasible library size for MicroBooNE light collection:
  - 25 x 25 x 100, 100,000 photons / vox ~ 10 x 10 x 10 cm voxels
- Scaling up, this is about 1600 hours of grid jobs.
- These numbers not optimized, and there may be room for increasing efficiency by disabling irrelevant physics in LArG4.

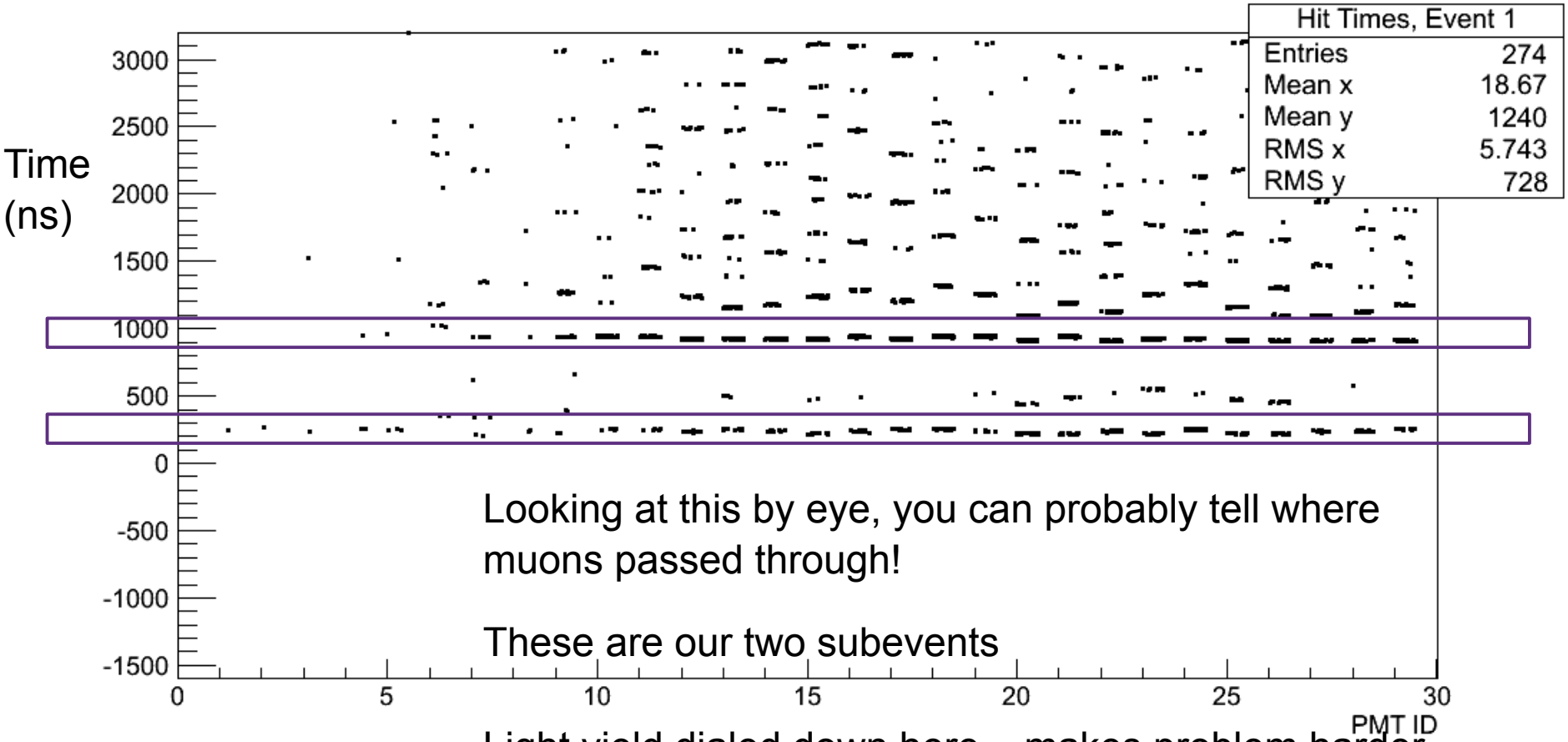
Geometrical T0 Finding

- **1: Find subevents by matching large PMT signals in time**
- **2: Make hypotheses of relative amount of light per PMT for each track**
- 3: Likelihood fit to match track to light hypothesis, and find T0.





Optical Subevent Finding



Looking at this by eye, you can probably tell where muons passed through!

These are our two subevents

Light yield dialed down here – makes problem harder, not easier

The algorithm

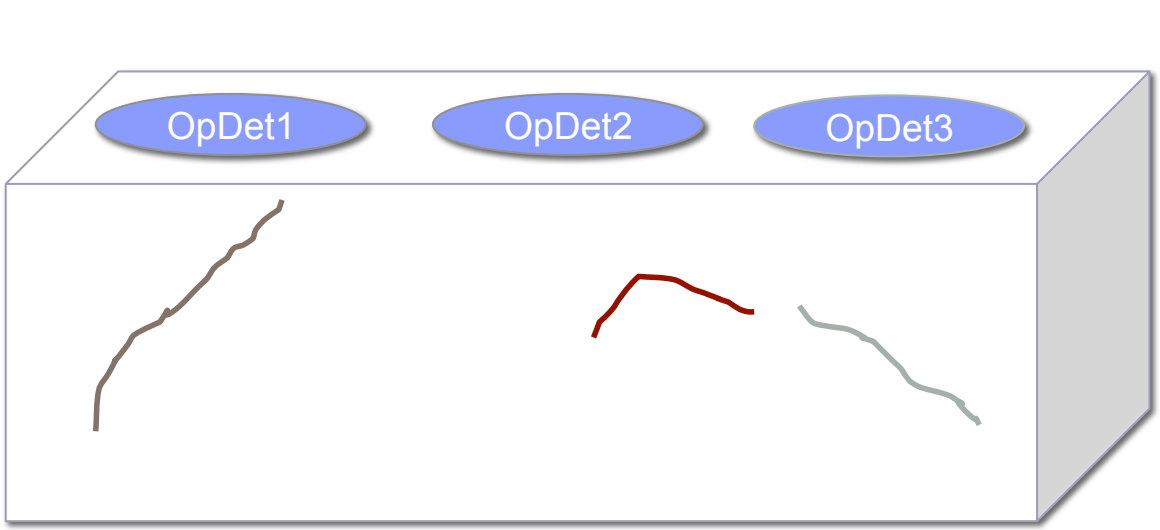
1) Compute average # & standard deviation of PE in peaks for each PMT
   - - We will only consider peaks larger than (avg – 1 stdev)
2) Find the largest peak
3) Scan through all peaks and find ones that fit in time with this largest peak
   - - Each peak can be in *at most* one subevent
4) Find the next largest peak not in a subevent, and repeat until all peaks have been sorted
5) Merge subevents that are very close in time and have completely disjoint sets of PMTs listed
   - - Each subevent has at most 1 peak from any given PMT

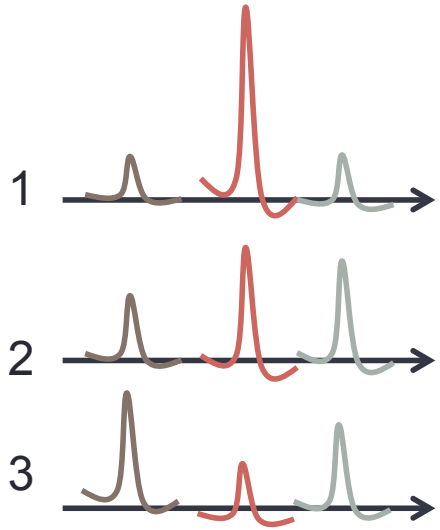
Accuracy

- Finds correct number of subevents for:
  - Up to 6 particles (muons), sometimes 7
  - Particle separation times > 180 ns
    - At smaller separation times, peaks merge
  - All these particles are traveling straight through detector volume (seen by most PMTs)
- Subevent timing fluctuates by about 100 ns (if we run same event multiple times)
  - Fit parameters waver slightly
- Now waiting on fast sim to be ready in order to test efficiency over different event classes – update next meeting

Geometrical T0 Finding

- **1: Find subevents by matching large PMT signals in time**
- **2: Make hypotheses of relative amount of light per PMT for each track**
- 3: Likelihood fit to match track to light hypothesis, and find T0.





Geometrical T0 Finding

- **1: Find subevents by matching large PMT signals in time**
- **2: Make hypotheses of relative amount of light per PMT for each track**
- 3: Likelihood fit to match track to light hypothesis, and find T0.

Part 2 ready to go – but ideally need library to be built to lookup visibility.

An undergrad, Deana Del Vecchio is going to help us assess the accuracy of using just 1/r^2 and solid angle to determine visibility

2. BEZIER TRACKS

Bezier Tracks – Module to Algorithm

- 1. Bezier tracker now updated to adhere to the algorithm / module scheme, like seeds, spacepoints etc.

- This partially so that bezier tracks can be used in interactive event display for handscan

Bezier Tracks - Calorimetry

- Bezier track now has a method to produce an anab::Calorimetry object.
- anab::Calorimetry modified to allow pitches for curved tracks, so one pitch per point. Previous constructors / interfaces are left unchanged for compatability.
- Pitch corrected dE/dx measurement in a given view, along curved track generated in BezierTrack method, not external calorimetry module, using Andrzejs's new "calo::CalorimetryAlg"
- Also interfaces with Mitches event display mode

root / trunk / TrackFinder / BezierTrack.h

History | View | Annotate | Download (3.2 kB)

```
anab::Calorimetry GetCalorimetryObject(std::vector<art::Ptr<recob::Hit> > Hits, geo::View_t view);
```

Bezier Tracks – Input modes

- Tracking module now has four modes:

- 1) Run using pre-generated seeeds
- 2) Run iteratively on unstructured hit collection
- 3) Single shot run 2D on cluster combinations
- 4) Iterative run on 2D cluster combinations (not tested)